

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of :
Takehiro YOSHIDA et al. :
Serial No. NEW : **Attn: APPLICATION BRANCH**
Filed April 1, 2004 : Attorney Docket No. 2004-0504A

PROGRAM LINKING PROGRAM, PROGRAM
PRODUCT, PROGRAM LINKING DEVICE,
TERMINAL DEVICE, AND PROGRAM
LINKING METHOD

THE COMMISSIONER IS AUTHORIZED
TO CHARGE ANY DEFICIENCY IN THE
FEES FOR THIS PAPER TO DEPOSIT
ACCOUNT NO. 23-0975

CLAIM OF PRIORITY UNDER 35 USC 119

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

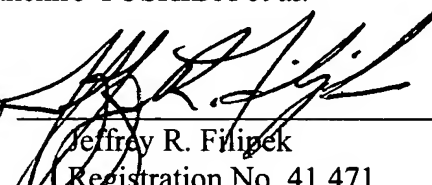
Applicants in the above-entitled application hereby claim the date of priority under the International Convention of Japanese Patent Application No. 2003-097892, filed April 1, 2003, as acknowledged in the Declaration of this application.

A certified copy of said Japanese Patent Application is submitted herewith.

Respectfully submitted,

Takehiro YOSHIDA et al.

By


Jeffrey R. Filipek
Registration No. 41,471
Attorney for Applicants

JRF/nk
Washington, D.C. 20006-1021
Telephone (202) 721-8200
Facsimile (202) 721-8250
April 1, 2004

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 4 月 1 日
Date of Application:

出 願 番 号 特 願 2 0 0 3 - 0 9 7 8 9 2
Application Number:
[ST. 10/C]: [J P 2 0 0 3 - 0 9 7 8 9 2]

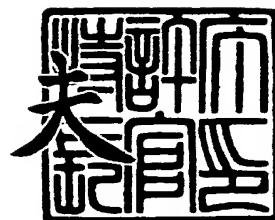
出 願 人 松 下 電 器 産 業 株 式 会 社
Applicant(s):



2 0 0 3 年 1 2 月 2 6 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康



【書類名】 特許願

【整理番号】 2968150023

【提出日】 平成15年 4月 1日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/445
G06F 13/00 354

【発明者】

【住所又は居所】 愛知県名古屋市中区栄2丁目6番1号白川ビル別館5階
株式会社松下電器情報システム名古屋研究所内

【氏名】 吉田 健宏

【発明者】

【住所又は居所】 愛知県名古屋市中区栄2丁目6番1号白川ビル別館5階
株式会社松下電器情報システム名古屋研究所内

【氏名】 川本 琢二

【発明者】

【住所又は居所】 愛知県名古屋市中区栄2丁目6番1号白川ビル別館5階
株式会社松下電器情報システム名古屋研究所内

【氏名】 河合 正樹

【特許出願人】

【識別番号】 000005821

【氏名又は名称】 松下電器産業株式会社

【代理人】

【識別番号】 100097445

【弁理士】

【氏名又は名称】 岩橋 文雄

【選任した代理人】

【識別番号】 100103355

【弁理士】

【氏名又は名称】 坂口 智康

【選任した代理人】

【識別番号】 100109667

【弁理士】

【氏名又は名称】 内藤 浩樹

【手数料の表示】

【予納台帳番号】 011305

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9809938

【書類名】 明細書

【発明の名称】 プログラムリンクプログラム、プログラムリンクプログラムを記録した記録媒体、プログラムリンク装置及びプログラムリンク方法

【特許請求の範囲】

【請求項 1】 複数のリンク前プログラムから 1 又は複数個をリンクしてリンク後プログラムを作成するステップと、

前記リンク後プログラムを記憶手段に記憶するステップと、

前記記憶手段の使用状況判断するステップとを有し、

前記記憶手段の使用状況判断に応じて、前記リンク後プログラムを作成するために必要なリンクの一部だけを行うか或いは全く行わないことがある、

プログラムリンクプログラム。

【請求項 2】 複数のリンク前プログラムから 1 又は複数個をリンクしてリンク後プログラムを作成するステップと、

前記リンク後プログラムを記憶手段に記憶するステップと、

前記記憶手段の使用状況判断するステップとを有し、

前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度で優先的にリンクを行う前記リンク前プログラムを決定するために、各リンク前プログラムがリンク後プログラム作成のために使われている頻度、或いは、各リンク前プログラムのサイズ、或いは、各リンク前プログラムがリンク後プログラム作成のために使われている頻度とそのリンク前プログラムサイズとの積、或いは、各リンク前プログラムの実行時リンク時間、或いは、各リンク前プログラムの実行頻度、の何れか 1 又は 2 以上を使用する、

プログラムリンクプログラム。

【請求項 3】 複数のリンク前プログラムから 1 又は複数個をリンクしてリンク後プログラムを作成するステップと、

前記リンク後プログラムを記憶手段に記憶するステップと、

前記記憶手段の使用状況判断するステップとを有し、

前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使

用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度で前記リンク後プログラムを作成するために、各リンク前プログラムがリンク後プログラム作成のために使われている頻度が低いものから優先的にリンクを行う

プログラムリンクプログラム。

【請求項 4】 複数のリンク前プログラムから 1 又は複数個をリンクしてリンク後プログラムを作成するステップと、

前記リンク後プログラムを記憶手段に記憶するステップと、

前記記憶手段の使用状況判断するステップとを有し、

前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度で前記リンク後プログラムを作成するために、各リンク前プログラムのサイズが小さなものから優先的にリンクを行う、

プログラムリンクプログラム。

【請求項 5】 複数のリンク前プログラムから 1 又は複数個をリンクしてリンク後プログラムを作成するステップと、

前記リンク後プログラムを記憶手段に記憶するステップと、

前記記憶手段の使用状況判断するステップとを有し、

前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度で前記リンク後プログラムを作成するために、各リンク前プログラムを実行時にリンクするために必要な時間が大きなものから優先的にリンクを行う、

プログラムリンクプログラム。

【請求項 6】 複数のリンク前プログラムから 1 又は複数個をリンクしてリンク後プログラムを作成するステップと、

前記リンク後プログラムを記憶手段に記憶するステップと、

前記記憶手段の使用状況判断するステップとを有し、

前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度

で前記リンク後プログラムを作成するために、各リンク前プログラムが実行される頻度の高いものから優先的にリンクを行う、

プログラムリンクプログラム。

【請求項 7】 複数のリンク前プログラムから 1 又は複数個をリンクしてリンク後プログラムを作成するステップと、

前記リンク後プログラムを記憶手段に記憶するステップと、

前記記憶手段の使用状況を判断するステップとを有し、

前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度で前記リンク後プログラムを作成するために、各リンク前プログラムがリンク後プログラム作成のために使われている頻度とそのリンク前プログラムサイズとの積が小さなものから優先的にリンクを行う、

プログラムリンクプログラム。

【請求項 8】 コンピュータ読み取り可能なデータ記録媒体であって、

請求項 1 から請求項 7 の何れか 1 項に記載のプログラムリンクプログラムを記録した記録媒体。

【請求項 9】 複数のリンク前プログラムから 1 又は複数個をリンクしてリンク後プログラムを作成するステップと、

前記リンク後プログラムを記憶手段に記憶するステップと、

前記記憶手段の使用状況を判断するステップとを有し、

前記記憶手段の使用状況の判断に応じて、前記リンク後プログラムを作成するために必要なリンクの一部だけを行うか或いは全く行わないことがある、

プログラムリンク方法。

【請求項 10】 複数のリンク前プログラムから 1 又は複数個をリンクしてリンク後プログラムを作成する手段と、

前記リンク後プログラムを記憶手段に記憶する手段と、

前記記憶手段の使用状況を判断する手段とを有し、

リンク後プログラムを作成する手段は、前記記憶手段の使用状況の判断に応じて、前記リンク後プログラムを作成するために必要なリンクの一部だけを行うか

或いは全く行わないことがある、

プログラムリンク装置。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本特許出願に係る発明は、いわゆるソースプログラム即ち原始言語で記述されたプログラムをコンパイラによって変換して得られる中間言語プログラム、具体的には例えば実行時の絶対アドレスが決定されていないプログラムや汎用のライブラリプログラムを複数結合し、例えば実行時の絶対アドレスが決定されたオブジェクトプログラムを作成するリンクに関するものである。

【0 0 0 2】

しかし上記、実行時の絶対アドレスの決定の有無は1つの例示であってこれに限定されるものではない。

【0 0 0 3】

特に、実行時環境でのメモリサイズに制限の大きい、携帯機器を始めとする各種装置で実行される、例えば J a v a (R) 言語で記述されたプログラムのリンクに関するものである。

【0 0 0 4】

しかしこれらもあくまで1つの例示であって、これに限定されるものではない。

【0 0 0 5】

【従来の技術】

従来の、J a v a (R) 言語やC言語で記述されたプログラムを複数結合し、実行プログラムを作成するリンクには、例えば、下記特許文献1、特許文献2に記載のものがあった。

【0 0 0 6】

例えば従来の J a v a (R) 実行環境では、アプリケーションプログラムの J a v a (R) ソースコードを中間言語のバイトコードに変換し、各クラス単位でファイルとして記憶される。

【0007】

アプリケーションが起動されると、これらのクラスファイルから必要なものをロードし、これらをリンクして実行プログラムが作成される。

【0008】

この方式を実行時リンク方式と呼び、この時のプログラムと処理の流れを図9に示す。

【0009】

図9に示す従来の実行時リンク方式のリンカでは、中間言語で記述されているクラスA、クラスB、クラスX、クラスYの各プログラムが例えばサーバ909に記憶されている。

【0010】

これらの4つのプログラムは事前に端末装置901にロードされ記憶される。

【0011】

例えばこれらの中からクラスA、クラスB、クラスXの各プログラムをリンクすることによってアプリケーションプログラム902が作成されると仮定する。

【0012】

このアプリケーションプログラム902が端末装置901で起動されると、その時点で実行時リンクが行われ、クラスA、クラスB、クラスXの各プログラムをリンクすることによってアプリケーションプログラム902が作成され、実行される。

【0013】

例えばこれらの中からクラスA、クラスB、クラスYの各プログラムをリンクすることによってアプリケーションプログラム903が作成されると仮定する。

【0014】

このアプリケーションプログラム903が端末装置901で起動されると、その時点で実行時リンクが行われ、クラスA、クラスB、クラスYの各プログラムをリンクすることによってアプリケーションプログラム903が作成され、実行される。

【0015】

又、例えば従来のC言語実行環境では、ソースプログラムをコンパイルしてオブジェクトプログラムに変換し、同時に必要な全てのライブラリを事前にリンクして実行形式ファイルを作成して記憶する。

【0016】

アプリケーションが起動されると、この実行形式ファイルがロードされ、実行される。

【0017】

この方式を事前リンク方式と呼び、この時のプログラムと処理の流れを図10に示す。

【0018】

図10に示す従来の事前リンク方式のリンカでは、中間言語で記述されているクラスA、クラスB、クラスX、クラスYの各プログラムが例えばサーバ929に記憶されていると同時に、サーバ929で事前リンクが行われ、アプリケーションプログラム922とアプリケーションプログラム923が作成されている。

【0019】

端末装置921はこれらのアプリケーションプログラム922とアプリケーションプログラム923をロードして記憶する。

【0020】

そしてこれらのアプリケーションプログラム922とアプリケーションプログラム923が起動されると、即時に実行することができる。

【0021】

【特許文献1】

特開平10-069376号公報

【特許文献2】

特表2000-514584号公報

【0022】

【発明が解決しようとする課題】

しかしながら例えば前記図9に示す実行時リンク方式では、端末装置901がロードして記憶するのは、クラスA、クラスB、クラスX、クラスYの各中間言

語プログラムを1つずつであるから、重複してライブラリを記憶する必要が無く、記憶領域を節減することができる。

【0023】

これは例えば携帯機器のように記憶部の容量に制限があるような機器では大きなメリットではあった。

【0024】

しかし一方、アプリケーションが起動されてからリンクを行うため、アプリケーションの起動から実際に実行されるまで長い時間を要すると言う欠点があった。

【0025】

又、例えば前記図10に示す事前リンク方式では、先にリンクを行って実行形式ファイルを作っておくため、アプリケーションが起動されるとすぐに実行することが可能であり、このことは大きなメリットである。

【0026】

しかし他方、事前にリンクを行っておくことによって、端末装置921は、アプリケーションプログラム922とアプリケーションプログラム923とで共通に使用するライブラリであるクラスAとクラスBを、重複して記憶しなければならない。

【0027】

これは例えば前記携帯機器のように記憶部の容量に制限があるような機器では大きな欠点であった。

【0028】

【課題を解決するための手段】

本特許出願の発明は、複数のリンク前プログラムから1又は複数個をリンクしてリンク後プログラムを作成するステップと、前記リンク後プログラムを記憶手段に記憶するステップと、前記記憶手段の使用状況を判断するステップとを有し、前記記憶手段の使用状況の判断に応じて、前記リンク後プログラムを作成するために必要なリンクの一部だけを行うか或いは全く行わないことがある、プログラムリンクプログラムである。

【0029】

本特許出願の発明は、複数のリンク前プログラムから1又は複数個をリンクしてリンク後プログラムを作成するステップと、前記リンク後プログラムを記憶手段に記憶するステップと、前記記憶手段の使用状況を判断するステップとを有し、前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度で優先的にリンクを行う前記リンク前プログラムを決定するために、各リンク前プログラムがリンク後プログラム作成のために使われている頻度、或いは、各リンク前プログラムのサイズ、或いは、各リンク前プログラムがリンク後プログラム作成のために使われている頻度とそのリンク前プログラムサイズとの積、或いは、各リンク前プログラムの実行時リンク時間、或いは、各リンク前プログラムの実行頻度、の何れか1又は2以上を使用する、プログラムリンクプログラムである。

【0030】

本特許出願の発明は、複数のリンク前プログラムから1又は複数個をリンクしてリンク後プログラムを作成するステップと、前記リンク後プログラムを記憶手段に記憶するステップと、前記記憶手段の使用状況を判断するステップとを有し、前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度で前記リンク後プログラムを作成するために、各リンク前プログラムがリンク後プログラム作成のために使われている頻度が低いものから優先的にリンクを行う、プログラムリンクプログラムである。

【0031】

本特許出願の発明は、複数のリンク前プログラムから1又は複数個をリンクしてリンク後プログラムを作成するステップと、前記リンク後プログラムを記憶手段に記憶するステップと、前記記憶手段の使用状況を判断するステップとを有し、前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度で前記リンク後プログラムを作成するために、各リンク前プログラムのサイズが

小さなものから優先的にリンクを行う、プログラムリンクプログラムである。

【0032】

本特許出願の発明は、複数のリンク前プログラムから1又は複数個をリンクしてリンク後プログラムを作成するステップと、前記リンク後プログラムを記憶手段に記憶するステップと、前記記憶手段の使用状況を判断するステップとを有し、前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度で前記リンク後プログラムを作成するために、各リンク前プログラムを実行時にリンクするために必要な時間が大きなものから優先的にリンクを行う、プログラムリンクプログラムである。

【0033】

本特許出願の発明は、複数のリンク前プログラムから1又は複数個をリンクしてリンク後プログラムを作成するステップと、前記リンク後プログラムを記憶手段に記憶するステップと、前記記憶手段の使用状況を判断するステップとを有し、前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度で前記リンク後プログラムを作成するために、各リンク前プログラムが実行される頻度の高いものから優先的にリンクを行う、プログラムリンクプログラムである。

【0034】

本特許出願の発明は、複数のリンク前プログラムから1又は複数個をリンクしてリンク後プログラムを作成するステップと、前記リンク後プログラムを記憶手段に記憶するステップと、前記記憶手段の使用状況を判断するステップとを有し、前記リンク後プログラムを作成するステップの実行によって前記記憶手段の使用状況が前記記憶手段の記憶容量を超える時には、前記記憶容量を超えない限度で前記リンク後プログラムを作成するために、各リンク前プログラムがリンク後プログラム作成のために使われている頻度とそのリンク前プログラムサイズとの積が小さなものから優先的にリンクを行う、プログラムリンクプログラムである。

。

【0035】

本特許出願の発明は、コンピュータ読み取り可能なデータ記録媒体であって、請求項1から請求項7の何れか1項に記載のプログラムリンクプログラムを記録した記録媒体である。

【0036】

本特許出願の発明は、複数のリンク前プログラムから1又は複数個をリンクしてリンク後プログラムを作成するステップと、前記リンク後プログラムを記憶手段に記憶するステップと、前記記憶手段の使用状況を判断するステップとを有し、前記記憶手段の使用状況の判断に応じて、前記リンク後プログラムを作成するために必要なリンクの一部だけを行うか或いは全く行わないことがある、プログラムリンク方法である。

【0037】

本特許出願の発明は、複数のリンク前プログラムから1又は複数個をリンクしてリンク後プログラムを作成する手段と、前記リンク後プログラムを記憶手段に記憶する手段と、前記記憶手段の使用状況を判断する手段とを有し、リンク後プログラムを作成する手段は、前記記憶手段の使用状況の判断に応じて、前記リンク後プログラムを作成するために必要なリンクの一部だけを行うか或いは全く行わないことがある、プログラムリンク装置である。

【0038】**【発明の実施の形態】**

本特許出願に係る発明（以後、本発明とも言う）の実施の形態を図面を使って詳細に説明する。

【0039】**（実施の形態1）**

本特許出願に係る発明（省略して「本発明」とも言う）であるプログラムリンクプログラム又はプログラムリンク方法（省略して単に「リンク」とも言う）を実行するプログラムリンク装置の一実施の形態である端末装置101のブロック構成図を図1に示す。

【0040】

この端末装置 101 とは更に具体的には、携帯電話や携帯情報端末（PDA：Personal Digital Assistants）等の携帯用情報機器、或いはパソコン等の情報機器、又はマイコンによって制御される種々の家庭用電気製品、事務用電子機器、その他制御装置等である。

【0041】

更に、他の機器、例えば図 1 のサーバ 109 から中間言語プログラムをダウンロードして実行する装置である。

【0042】

但しこのサーバ 109 は 1 つの例であって、必ずしもサーバ 109 に限るものではなく、他の種々の電子機器、例えば通常のパソコンや電気製品であっても良い。

【0043】

この端末装置 101 で主にリンクを実行するのは事前リンク部 141 である。

【0044】

この事前リンク部 141 の作用を中心に、図 1、図 2、図 3、図 4、図 5 を使って説明する。

【0045】

この端末装置 101 では様々なアプリケーションが実行される。例えば図 4、図 5 に示すアプリケーション 1、アプリケーション 2、アプリケーション 3 が実行され、これを実際に実行するのはアプリケーションプログラム 1、アプリケーションプログラム 2、アプリケーションプログラム 3 である。

【0046】

特に区別する必要が無い時には、アプリケーションプログラム 1、アプリケーションプログラム 2、アプリケーションプログラム 3 と、アプリケーション 1、アプリケーション 2、アプリケーション 3 との両方の意味を含めて、単にアプリケーション 1、アプリケーション 2、アプリケーション 3 とも言う。

【0047】

又同様に、特に区別する必要が無い時には、アプリケーションプログラムとアプリケーションとの両方の意味を含めて、単にアプリケーションとも言う。

【0048】

これらのアプリケーションプログラムは、幾つかの中間言語で記述されたクラスライブラリから構成されており、この中間言語のクラスライブラリはサーバ109に記憶されている。

【0049】

図1のサーバ109にはクラスA、クラスB、クラスX、クラスYの4つのクラスだけが記載されているが、勿論これらだけに限るものではなく、その他必要となる全てのクラスライブラリが記憶されているが、図1では記載を省略している。

【0050】

端末装置101の制御部131でダウンロード命令113が実行されると、サーバ109から端末装置101にこれらのクラスライブラリがダウンロードされる。

【0051】

このダウンロードは図示していないが、種々のネットワーク、例えばインターネットやLAN、CATV、衛星回線、電話回線、ISDN、ATM、その他のネットワークを経由してダウンロードされることもあり、又一方いわゆるパッケージ形態、即ち何等かのパッケージメディアに記録された状態で端末装置101に読み込まれることもある。

【0052】

又、ダウンロードの指定方法には、端末装置101が実行するアプリケーションの中の特定のアプリケーションが指定され、その指定されたアプリケーションの実行のために必要なクラスライブラリだけをダウンロードする方法と、端末装置101が実行し得る全てのアプリケーションの実行のために必要なクラスライブラリをダウンロードする方法と、サーバ109が保有している全てのクラスライブラリをダウンロードする方法と、その他、種々の方法でクラスライブラリが指定されその指定されたクラスライブラリがダウンロードされる方法、その他がある。

【0053】

このダウンロードを行うのはダウンロード制御部 133 であり、ダウンロードされたクラスライブラリはファイルシステム部 137 に記憶保存される。

【0054】

制御部 131 で事前リンク命令 115 が実行されると、ファイルシステム部 137 に記憶保存されているダウンロードされたクラスライブラリの事前リンクが行われる。

【0055】

前記の通り、各アプリケーションの実行に先立って事前にリンクしておくことにより、実行時の速度を向上するためである。

【0056】

又、この事前リンク命令の実行は、ダウンロードのタイミングに同期して、或いはダウンロードのタイミングと同時に行われても良いし、ある所定の時間によって行われても良いし、ユーザの何等かの操作に対応して行われても良いし、その他の何等かのタイミングによって行われても良い。

【0057】

事前リンク命令 115 が実行されると制御部 131 から事前リンク部 141 に対して事前リンク実行の制御が行われ、以下の処理がスタートする（図 2、220）。

【0058】

事前リンク部 141 では、まず最初に全てのクラスライブラリのリンクが行われる（図 2、221）。

【0059】

これはクラス呼び出し関係検出部 146 が、各アプリケーション毎に呼び出しているクラスライブラリを検出し、更に呼び出されているクラスライブラリがまた呼び出しているクラスライブラリを検出し、この処理を繰り返すことによって行われる。

【0060】

呼び出されていることが検出されたクラスライブラリは結合部 143 によって結合（リンク）される。この結合（リンク）は実行時アドレスの決定を伴うこと

もあるし、必ずしも実行時アドレスの最終的な決定まで到らないこともあり、どちらかに限るものではない。

【0061】

このリンク処理を実行している時、メモリ量管理部144は各リンクの実行によってファイルシステム部137のメモリ量（残り容量）がどのように変化するかを管理している。

【0062】

以上の処理を何回か必要な回数繰り返すことによって、端末装置101が実行できる、或いは実行しなければならない、或いは指定された、全てのアプリケーションプログラムのリンクが終了する（図2、223）。

【0063】

この時点で、メモリ量管理部144は、管理しているファイルシステム部137のメモリ量（残り容量）を確認し、その容量内に収まっているか、或いはメモリオーバーフローしているかを判断する（図2、225）。

【0064】

もしもファイルシステム部137のメモリ容量内に収まっているならば、全てのアプリケーションプログラムが正常にリンクできたことになり、事前リンクは終了する（図2、228）。

【0065】

もしもファイルシステム部137のメモリ容量をオーバーフローしているならば、リンク数検出部142が、現在リンクに使われているクラスライブラリの中で、最も多くのアプリケーションのために使用されているクラスライブラリを検出し、そのクラスライブラリを使用しているアプリケーション数（最大値に相当）を求める（図2、231）。説明の都合上、その数を仮にNとする（図2、231）。

【0066】

そして当初は「全てのクラスライブラリのリンクを行う」と設定されていたのが、これ以降は「N個以上のアプリケーションで使用されているクラスライブラリのリンクは行わない」と、設定の変更が行われる（図2、233）。

【0067】

そして一旦既に行った全てのアプリケーションプログラムのリンクを解除し、再び新しい設定に従って全てのアプリケーションのリンクを行う（図2、223）。

【0068】

尚ここで、一旦既に行った全てのアプリケーションのリンクを解除する方法として、リンク済みのプログラムのリンクを分解する方法や、リンク前のプログラムを別に保存しておく方法、その他の方法が有り得るが、どのような方法を使用しても構わない。

【0069】

例えばリンク前のクラスライブラリをダウンロード制御部133に保存しておく方法も有り得るし、例えば再度ダウンロード制御部133がサーバ109からリンク前のクラスライブラリをダウンロードする方法も有り得る。

【0070】

或いは結合ファイル解除部135がリンク済みのプログラムのリンクを分解し、再度ファイルシステム部137に渡す方法も有り得る。これらの何れの方法を使用しても構わない。

【0071】

例えば前記の通り、この端末装置101で実行されるアプリケーションが、例えば図4、図5に示すアプリケーション1、アプリケーション2、アプリケーション3であったと仮定する。

【0072】

このケースでは、全アプリケーションの実行に必要なクラスライブラリは、X、Y、A、B、C、D、Zであり（省略してクラスライブラリを特定する記号だけを記している、以後同じ）、これらがサーバ109からダウンロードされ記憶保存されている。

【0073】

全アプリケーションプログラムのリンクが実行されることによって（図2、223）、図4に示すステップ1、ステップ2、ステップ3が実行され、端末装置

101に全クラスライブラリがリンクされたアプリケーションプログラム1、アプリケーションプログラム2、アプリケーションプログラム3が作成される（図4、図5、ステップ3）。

【0074】

この状態では、全てのクラスライブラリが既にリンクされているので、図1の実行時リンク部152が実行時に改めてリンクを行うことは必要でなく、実行時のリンク数は0である（図5、ステップ3）。

【0075】

しかし一方、重複を含めずに数えた全クラスライブラリ数は上記7であるが、重複を含めて数えた全クラスライブラリ数は11であり、4つのクラスライブラリが重複して記憶されている（図5、ステップ3）。この重複する4つのクラスライブラリとは2つのDと、1つずつのBとCである。

【0076】

この重複する4つのクラスライブラリの分だけ、ファイルシステム部137のメモリ容量をオーバーフローする可能性が高くなっている。

【0077】

そこでこれらクラスライブラリの中で、最も多くのアプリケーションプログラムで使用されているものが、リンク数検出部142によって検出される（図2、231）。

【0078】

今の例では、このクラスライブラリはDであり、3つのアプリケーション全てで使用されている（図5、ステップ4）。従って、上記Nは3になる。

【0079】

そこで以降は、「3個以上のアプリケーションで使用されているクラスライブラリはリンクしない」と設定し直され（図2、233）、改めて全てのリンクが行われる（図2、223）。

【0080】

尚、図4のステップ4では、クラスライブラリDのリンクを解除するように記載している通り、必ずしも一旦全てのクラスライブラリのリンクを解除してリン

クし直す方法に限るものではなく、例えば3個以上のアプリケーションで使用されているクラスライブラリだけを検出しながらそのリンクを解除する方法でも構わない。

【0081】

又、図2に示す上記の方法では一旦全てのアプリケーションのリンクを実行してからメモリ容量のオーバーフローを検出しているが、図3には1つのアプリケーションのリンクを行う毎にメモリ容量のオーバーフローを検出する方法を示している。どちらの方法も本質的な部分は同一であるから、後者の方法については詳細な説明を省略する。

【0082】

このようにクラスライブラリDのリンクを解除した状態では、アプリケーションプログラム1～3の全てで実行時にクラスライブラリDのリンクを実行することが必要になるので、必要な実行時リンク数は3である（図5、ステップ4）。

【0083】

そのためにある程度実行時の速度は低下するが、その一方、重複を含めて数えた全クラスライブラリ数は9であり（図5、ステップ4）、重複して記憶されているクラスライブラリ数は2に減少している。その分、ファイルシステム部137がメモリ容量をオーバーフローする可能性は低くなる。この重複する2つのクラスライブラリとは、1つずつのBとCである。

【0084】

従って、図2の225では、今回の2周目は前回よりもメモリオーバーフローする可能性は低くなる。

【0085】

そこでメモリオーバーフローが発生しなければ、終了する（図2、228）。

【0086】

しかしこれでもファイルシステム部137がメモリ容量をオーバーフローしたと仮定する（図2、225）。このケースは今の例では、図2の225が2周目の判断でもメモリオーバーフローを検出した時に相当する。

【0087】

このケースでは再び図2の231で、現在リンクされているクラスライブラリの中で、最も多くのアプリケーションによって使用されているものが、リンク数検出部142によって検出される。

【0088】

それは図5のステップ5に示す通り、BとCであり、これらは共に2つのアプリケーションで使用されている。

【0089】

従って次に、「2個以上のアプリケーションで使用されているクラスライブラリはリンクしない」と再設定される（図2、233、2周目）。

【0090】

そして再び一旦全てのリンクが解除された後、図2の223で3度目のリンクが実行される。

【0091】

念のため説明すると、1周目のリンクでは全てのクラスライブラリのリンクが行われ、2周目のリンクでは3個以上のアプリケーションで使用されているクラスライブラリのリンクが行われず、今回の3周目では2個以上のアプリケーションで使用されているクラスライブラリのリンクが行われない。

【0092】

或いは、2個以上のアプリケーションで使用されているクラスライブラリのリンクが解除されるとしても良い。

【0093】

この状態を図5のステップ5に示す。

【0094】

この状態では、クラスライブラリD、B、Cのリンクが行われていないので、アプリケーションの実行時にはこれらのクラスライブラリから必要なクラスライブラリのリンクを行うことが必要であり、その総数は7である。

【0095】

何故ならば、アプリケーション1の実行時には、Xのクラスライブラリを基準に考えると、このXに対してB、C、Dの3つをリンクする必要がある、アプリ

ケーション2の実行時には、YとAのクラスライブラリは既にリンクされているが、このリンクされたものに対してB、Dの2つをリンクする必要がある、アプリケーション3の実行時には、Zのクラスライブラリを基準に考えると、このZに対してC、Dの2つをリンクする必要があるからである。

【0096】

従って、実行時の速度はその分低下するが、少なくともアプリケーション2で必要なクラスライブラリYとAとが既にリンクされているだけ何も事前リンクされていないよりも実行時の速度は向上する。

【0097】

一方、重複して記憶されているクラスライブラリは存在しない。従ってメモリ容量の無駄はその意味では全く生じていない。

【0098】

このことは、前記「2個以上のアプリケーションで使用されているクラスライブラリはリンクしない」と言う条件は、「唯1つのアプリケーションで使用されているクラスライブラリしかリンクしない」ことを意味し、この条件では明らかにクラスライブラリを重複して記憶することは生じ得ないことから理解することができる。

【0099】

図1の端末装置101の制御部において、実行命令117が実行されると制御部131から実行制御部151に対してアプリケーションの実行が指示される。

【0100】

この実行命令117の実行はユーザの何等かの操作や、指示や、他の機器からの制御や、時間その他の起動による場合等がある。

【0101】

又、通常このアプリケーションの起動はどのアプリケーションを起動するのか、例えば今までの例では、アプリケーション1か、アプリケーション2か、アプリケーション3なのかを指定して行われる。

【0102】

実行制御部151に特定の指定されたアプリケーションの実行が指示されると

、実行制御部 151 の実行時リンク部 152 がそのアプリケーションを実行するためにまだ幾らかのクラスライブラリのリンクすることが必要であるか否かを判断し、もしも必要であるならばそのクラスライブラリのリンクを実行する。

【0103】

例えば、図 5 のステップ 3 の状態であるならば、どのアプリケーションの実行が指示されたとしても、実行時にリンクしなければならないクラスライブラリは存在しない。

【0104】

例えば、図 5 のステップ 4 の状態で、アプリケーション 1 ～ 3 の何れかの実行が指示されたとすると、クラスライブラリ D の実行時リンクが必要になり、実行時リンク部 152 はこのクラスライブラリ D のリンクを実行する。

【0105】

例えば、図 5 のステップ 5 の状態で、アプリケーション 1 の実行が指示されたとすると、クラスライブラリ B と C と D の実行時リンクが必要となり、実行時リンク部 152 はこのクラスライブラリ B と C と D のリンクを実行する。

【0106】

例えば、図 5 のステップ 5 の状態で、アプリケーション 2 の実行が指示されたとすると、クラスライブラリ B と D の実行時リンクが必要となり、実行時リンク部 152 はこのクラスライブラリ B と D のリンクを実行する。

【0107】

例えば、図 5 のステップ 5 の状態で、アプリケーション 3 の実行が指示されたとすると、クラスライブラリ C と D の実行時リンクが必要となり、実行時リンク部 152 はこのクラスライブラリ C と D のリンクを実行する。

【0108】

このようにして実行に必要な全てのクラスライブラリのリンクが行われると、実行部 153 がそのアプリケーションプログラムの実行を行う。

【0109】

(実施の形態 2)

次に本発明の第 2 の実施の形態であるプログラムリンクプログラム、プログラ

ムリンク方法、プログラムリンク装置について説明する。但し、前記第1の実施の形態と本質的に異なる部分についてのみ説明し、前記第1の実施の形態と本質的に同一部分については説明を省略する。

【0110】

この第2の実施の形態でも、全てのアプリケーションプログラムのリンクを行って、ファイルシステム部137のメモリ容量内に収まっているならば、事前リンクは正常に終了する。

【0111】

このことに関して、本発明の第1の実施の形態と同じである。

【0112】

もしもファイルシステム部137のメモリ容量をオーバーフローするならば、クラスライブラリの中で、より少ないアプリケーションのためにリンクされているクラスライブラリから優先的にリンクを行う。このことに関して、本発明の第1の実施の形態と同じである。

【0113】

この第2の実施の形態が前記第1の実施の形態と本質的に異なる部分は、より少ないアプリケーションのためにリンクされているクラスライブラリから優先的にリンクを行うための手順に関してのみである。

【0114】

従って、この手順についてのみ図6を参照して説明する。

【0115】

前記第1の実施の形態では、全てのリンクを実際に行うことによってファイルシステム部137のメモリ容量をオーバーフローするか否かを判定した。しかしメモリ容量の判定のために実施のリンクを行うことは非効率な部分を含む。

【0116】

そこでこの第2の実施の形態では、実際にリンクを行う前に、メモリサイズのチェックを行い、ファイルシステム部137のメモリに収容可能な最大限のライブラリを上記優先基準に基づいて予め求め、実際のリンクはそのライブラリに関してのみ、その後に行う。

【0117】

図6はそのための手順を示している。

【0118】

この実施の形態では、まず、各ライブラリ毎に、そのライブラリがアプリケーションからリンクされる数、即ち、そのライブラリをリンクしているアプリケーションの数を「共有数」と定義し、この値を説明のため(n)と記して、この(n)を各ライブラリ毎に調べることによって、求める(図6、622)。

【0119】

次に、全ライブラリをこの共有数(n)でソートし、(n)の値が小さなものから順に並べ、0(ゼロ)から始まるインデックス番号を付し、テーブルを作成する(図6、623)。

【0120】

同じ(n)の値となるライブラリの順序についてはどのように設定しても構わないし、或いは他の何等かの方法によって順序を設定しても構わない。例えば1つの方法として、同じ(n)の値となるライブラリの順序については後述する本発明の第3の実施の形態から第6の実施の形態で説明する判断基準を使用することができる。

【0121】

次に、処理のために必要な2つの変数として「インデックス変数」と「リンク総和量」とを用意し、それぞれ0(ゼロ)に初期化する(図6、624)。

【0122】

ここからループの処理に入り、最初に、前記インデックス変数の値が、ライブラリの総数と等しいか否かを判断する(図6、625)。

【0123】

これはメモリをオーバーフローすることなく、全てのライブラリのリンクが完了したか否かを判断するためである。

【0124】

即ち、メモリをオーバーフローすることなく、全てのライブラリのリンクが完了すれば、この判定で、前記インデックス変数の値がライブラリの総数と等しく

なる。

【0125】

しかし最初の段階では、インデックス変数の値はゼロで、ライブラリ総数は1以上であるから等しくはならない。

【0126】

そこで前記インデックス番号の値がインデックス変数の値に等しいライブラリの増加量をリンク総和量に加える（図6、626）。

【0127】

ここで各ライブラリの増加量とは、図6には明記していないが、そのライブラリのメモリサイズとそのライブラリをリンクしているアプリケーション数マイナス1との積である（図7、722参照）。

【0128】

即ち、このライブラリの増加量とは、そのライブラリをリンクすることによって新たに必要となるメモリ容量（必要メモリの増加量）に相当する。

【0129】

この段階で、リンク総和量が、ファイルシステム部137のメモリ容量をオーバーフローするか否かを判定する（図6、627）。

【0130】

もしもこの段階で、リンク総和量がファイルシステム部137のメモリ容量をオーバーフローするならば、現在のインデックス変数に相当するインデックス番号のライブラリをリンクするとメモリオーバーフローすることになるので、その1つ手前、即ちインデックス番号が現在のインデックス変数の値マイナス1のライブラリまでのリンクを実行する（図6、629）。

【0131】

もしもこの段階で、リンク総和量がファイルシステム部137のメモリ容量をオーバーフローしないならば、現在のインデックス変数に相当するインデックス番号のライブラリをリンクしてもメモリオーバーフローを生じないことが解る。

【0132】

そこで次にインデックス変数を1増加し（図6、628）、これによって全て

のライブラリに関してリンク可否の確認が終了するかの判断に戻る（図 6、625）。これは前記処理ループの先頭である。

【0133】

これによってインデックス変数値が全ライブラリ数と等しくなれば、全てのライブラリの確認が終り、全ライブラリがリンク可能であることになり、全ライブラリのリンクを実行することができる（図 6、629）。

【0134】

全ライブラリの確認が終わっていなければ、次のライブラリの増加量をリンク総和量に加え（図 6、626）、リンク可能か否かの判断を繰り返す（図 6、627）。

【0135】

この第 2 の実施の形態が前記第 1 の実施の形態と本質的に異なる部分は、以上説明した通り、リンクを実際に実行する前に、リンク可能なライブラリの範囲を確定することとそのための手順だけであるので、他に関しては説明を省略する。

【0136】

（実施の形態 3）

次に本発明の第 3 の実施の形態であるプログラムリンクプログラム、プログラムリンク方法、プログラムリンク装置について説明する。但し、前記第 2 の実施の形態と本質的に異なる部分についてのみ説明し、前記第 2 の実施の形態と本質的に同一部分については説明を省略する。

【0137】

この第 3 の実施の形態でも、前記第 2 の実施の形態と同様に、予めライブラリを所定の順序に従って並べ、その順に増加量をリンク総和量に加えることによってそのライブラリがリンク可能か否かを判断することに関し、前記第 2 の実施の形態と同じである。

【0138】

そのための処理手順を図 7 に示す。

【0139】

この図 7 に示す本発明の第 3 の実施の形態の処理手順に関し、前記図 6 に示す

本発明の第2の実施の形態の処理手順と本質的に相違するのは、予め全ライブラリを増加量でソートし、この増加量が小さなものから順にインデックス番号を付すことのみである（図7、723）。

【0140】

この増加量とは、前記の通り、各ライブラリ毎にそのライブラリがアプリケーションからリンクされている数（共有数（ n ））マイナス1とそのライブラリのサイズとの積である（図7、722）。

【0141】

これは前記の通り、そのライブラリをリンクすることによって新たに必要となるメモリの増加量に相当する。

【0142】

そしてインデックス番号順に1つずつライブラリをリンクすることによってメモリオーバーフローを生じることが無いか否かを判断することは、前記第2の実施の形態と同じであるが（図7、725～729）、この第3の実施の形態ではインデックス番号が増加量の小さな順に付されているため、この増加量の順にライブラリのリンクの可否を判断し、可能な限り増加量が小さなライブラリからリンクが実行されることについてのみ、前記第2の実施の形態と相違する。

【0143】

この第3の実施の形態が前記第2の実施の形態と本質的に異なる部分は、以上説明した部分だけであるので、他に関しては説明を省略する。

【0144】

（実施の形態4）

次に本発明の第4の実施の形態であるプログラムリンクプログラム、プログラムリンク方法、プログラムリンク装置について説明する。但し、前記第2の実施の形態と本質的に異なる部分についてのみ説明し、前記第2の実施の形態と本質的に同一部分については説明を省略する。

【0145】

この第4の実施の形態でも、前記第2の実施の形態と同様に、予めライブラリを所定の順序に従って並べ、その順に増加量をリンク総和量に加えることによ

てそのライブラリがリンク可能か否かを判断することに関し、前記第2の実施の形態と同じである。

【0146】

そのための処理手順を図8に示す。

【0147】

この図8に示す本発明の第4の実施の形態の処理手順に関し、前記図6に示す本発明の第2の実施の形態の処理手順と本質的に相違するのは、予め全ライブラリについてそのライブラリがリンクされているアプリケーションを調べる（図8、821）のと同時に、そのアプリケーションが使われる頻度を調べ（図8、822）、全ライブラリをこの使用頻度でソートし、この使用頻度が大きなものから順にインデックス番号を付すことのみである（図8、823）。

【0148】

例えば、あるライブラリが電子メール閲覧アプリケーションソフトによってリンクされているならば、その電子メール閲覧アプリケーションソフトの使用頻度は比較的高い、例えば1日平均5回程度であるかも知れない。

【0149】

或いは、あるライブラリが銀行振込アプリケーションソフトによってリンクされているならば、その銀行振込アプリケーションソフトの使用頻度は比較的低い、例えば平均1年間に数回程度であるかも知れない。

【0150】

例えばこのようなケースでは、前記電子メール閲覧アプリケーションソフトによってリンクされているライブラリの方が、銀行振込アプリケーションソフトによってリンクされているライブラリよりも小さなインデックス番号が付され、優先的にリンクが実行される。

【0151】

又、例えば1つのライブラリが複数のアプリケーションによってリンクされていることもある。例えば、ある比較的高頻度の高いライブラリは前記電子メール閲覧アプリケーションソフトと銀行振込アプリケーションソフトの両方でリンクされているかも知れない。

【0152】

このようなライブラリに関しては、そのライブラリをリンクしているアプリケーションソフトの使用頻度をどのようにして求めるかに関しては種々の方法が考えられる。

【0153】

例えば、そのライブラリをリンクしているアプリケーションソフトの中で最も使用頻度の高いものを用いても良いし、平均値を用いても良いし、中央値を用いることもできるし、総和を用いても構わない。

【0154】

最も使用頻度の高いアプリケーションソフトの使用頻度或いは、そのライブラリをリンクしている全アプリケーションソフトの使用頻度の総和を用いることが適当かも知れないが、その方法に限定されるものではない。

【0155】

このようにして、ライブラリをリンクしているアプリケーションソフトの使用頻度を基準に、リンクを実行するライブラリを決定することによって、比較的使用頻度の高いアプリケーションソフトの実行速度を向上させることができ、より使い勝手を良くすることができる。

【0156】

インデックス番号順に1つずつライブラリをリンクすることによってメモリオーバーフローを生じることが無いか否かを判断することは、前記第2の実施の形態と同じである（図8、825～829）。

【0157】

この第4の実施の形態が前記第2の実施の形態と本質的に異なる部分は、以上説明した部分だけであるので、他に関しては説明を省略する。

【0158】

（実施の形態5）

次に本発明の第5の実施の形態であるプログラムリンクプログラム、プログラムリンク方法、プログラムリンク装置について説明する。但し、前記第2の実施の形態と本質的に異なる部分についてのみ説明し、前記第2の実施の形態と本質

的に同一部分については説明を省略する。

【0159】

この第5の実施の形態でも、前記第2の実施の形態と同様に、予めライブラリを所定の順序に従って並べ、その順に増加量をリンク総和量に加えることによってそのライブラリがリンク可能か否かを判断することに関し、前記第2の実施の形態と同じである。

【0160】

この第5の実施の形態の処理手順に関し、本発明の第2の実施の形態の処理手順と本質的に相違するのは、予め全ライブラリをそのサイズによってソートし、そのサイズの小さな順にインデックス番号を付し、このインデックス番号順にリンクが可能か否かを判断することについてのみである。

【0161】

このようにすることによって、比較的メモリサイズの小さなライブラリを優先的にリンクすることができる。

【0162】

この第5の実施の形態が前記第2の実施の形態と本質的に異なる部分は、以上説明した部分だけであるので、他に関しては説明を省略する。

【0163】

(実施の形態6)

次に本発明の第6の実施の形態であるプログラムリンクプログラム、プログラムリンク方法、プログラムリンク装置について説明する。但し、前記第2の実施の形態と本質的に異なる部分についてのみ説明し、前記第2の実施の形態と本質的に同一部分については説明を省略する。

【0164】

この第6の実施の形態でも、前記第2の実施の形態と同様に、予めライブラリを所定の順序に従って並べ、その順に増加量をリンク総和量に加えることによってそのライブラリがリンク可能か否かを判断することに関し、前記第2の実施の形態と同じである。

【0165】

この第6の実施の形態の処理手順に関し、本発明の第2の実施の形態の処理手順と本質的に相違するのは、予め全ライブラリをその実行時リンクに必要な時間によってソートし、その時間が大きなものから順にインデックス番号を付し、このインデックス番号順にリンクが可能か否かを判断することについてのみである。

【0166】

このようにすることによって、実行時リンクに必要な時間が大きなライブラリを優先的にリンクすることができ、全体としての実行速度を改善することができる。

【0167】

この第6の実施の形態が前記第2の実施の形態と本質的に異なる部分は、以上説明した部分だけであるので、他に関しては説明を省略する。

【0168】

【発明の効果】

以上詳細に説明したように、本発明によるならば、端末装置のメモリ容量を最大限に使用しながら、また同時にアプリケーション実行時の速度を最大限向上させることができ、その実用的効果は極めて大きい。

【0169】

またこの効果は特にメモリ容量に制限の大きな携帯機器では更に大きなものとなる。

【図面の簡単な説明】

【図1】

本発明の第1の実施の形態のリンクを実行する端末装置のブロック構成図

【図2】

本発明の第1の実施の形態のリンクを実行する端末装置の処理流れ図

【図3】

本発明の第1の実施の形態のリンクの変形を実行する端末装置の処理流れ図

【図4】

本発明の第1の実施の形態のリンクが実行される過程のクラスライブラリの構

造変化を示す図

【図 5】

本発明の第 1 の実施の形態のリンクが実行される過程のクラスライブラリ数とリンク数を示す図

【図 6】

本発明の第 2 の実施の形態のリンクを実行する端末装置の処理流れ図

【図 7】

本発明の第 3 の実施の形態のリンクを実行する端末装置の処理流れ図

【図 8】

本発明の第 4 の実施の形態のリンクを実行する端末装置の処理流れ図

【図 9】

従来の J a v a (R) リンカの実行例（実行時リンク方式）を示す図

【図 10】

従来の C リンカの実行例（事前リンク方式）を示す図

【符号の説明】

101, 901, 921 端末装置

109, 909, 929 サーバ

113 ダウンロード命令

115 事前リンク命令

117 実行命令

131 制御部

133 ダウンロード制御部

135 結合ファイル解除部

137 ファイルシステム部

141 事前リンク部

142 リンク数検出部

143 結合部

144 メモリ量管理部

146 クラス呼び出し関係検出部

1 5 1 実行制御部

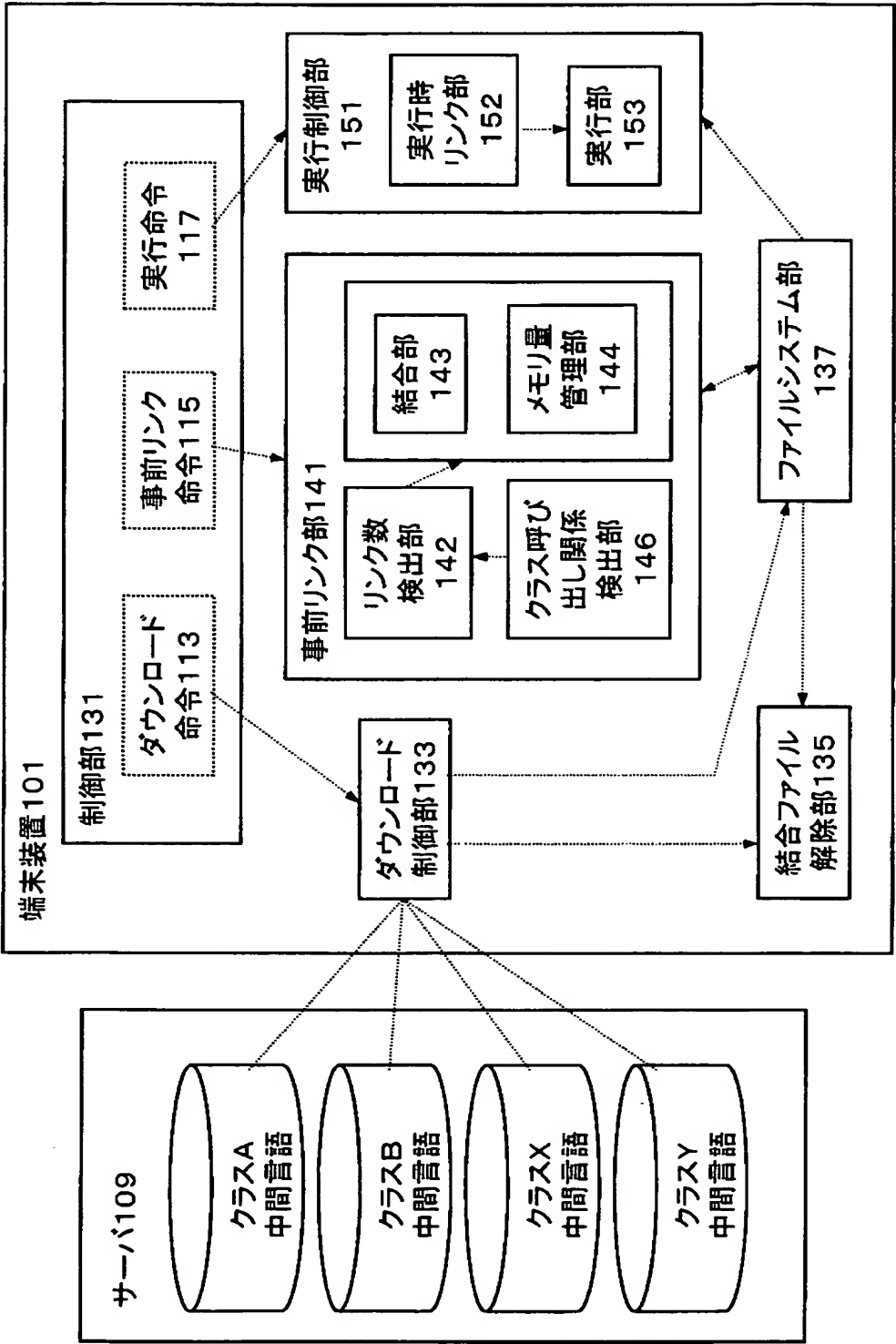
1 5 2 実行時リンク部

1 5 3 実行部

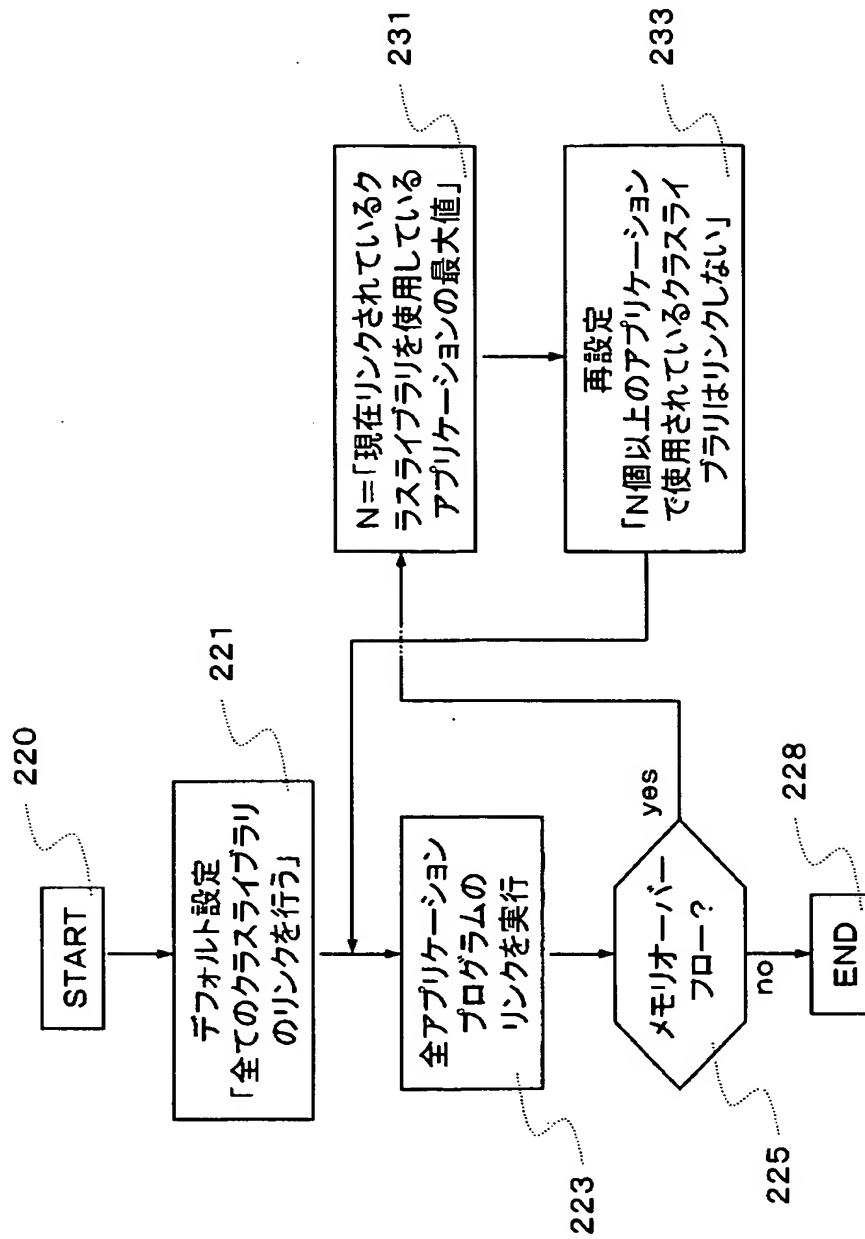
9 0 2, 9 0 3, 9 2 2, 9 2 3 アプリケーションプログラム

【書類名】 図面
【図 1】

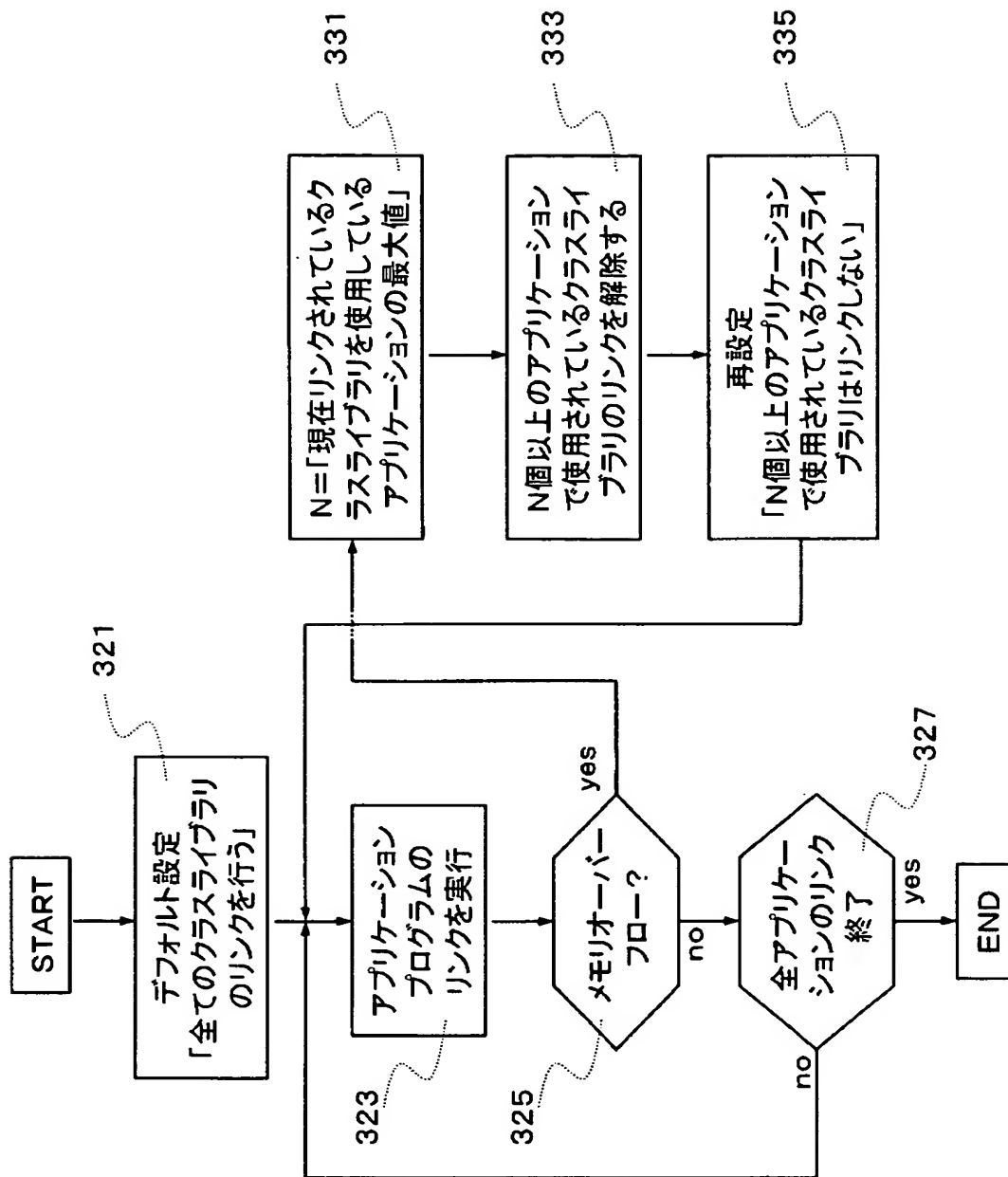
本発明の第1の実施の形態の形態のリンクを実行する端末装置のブロック構成図



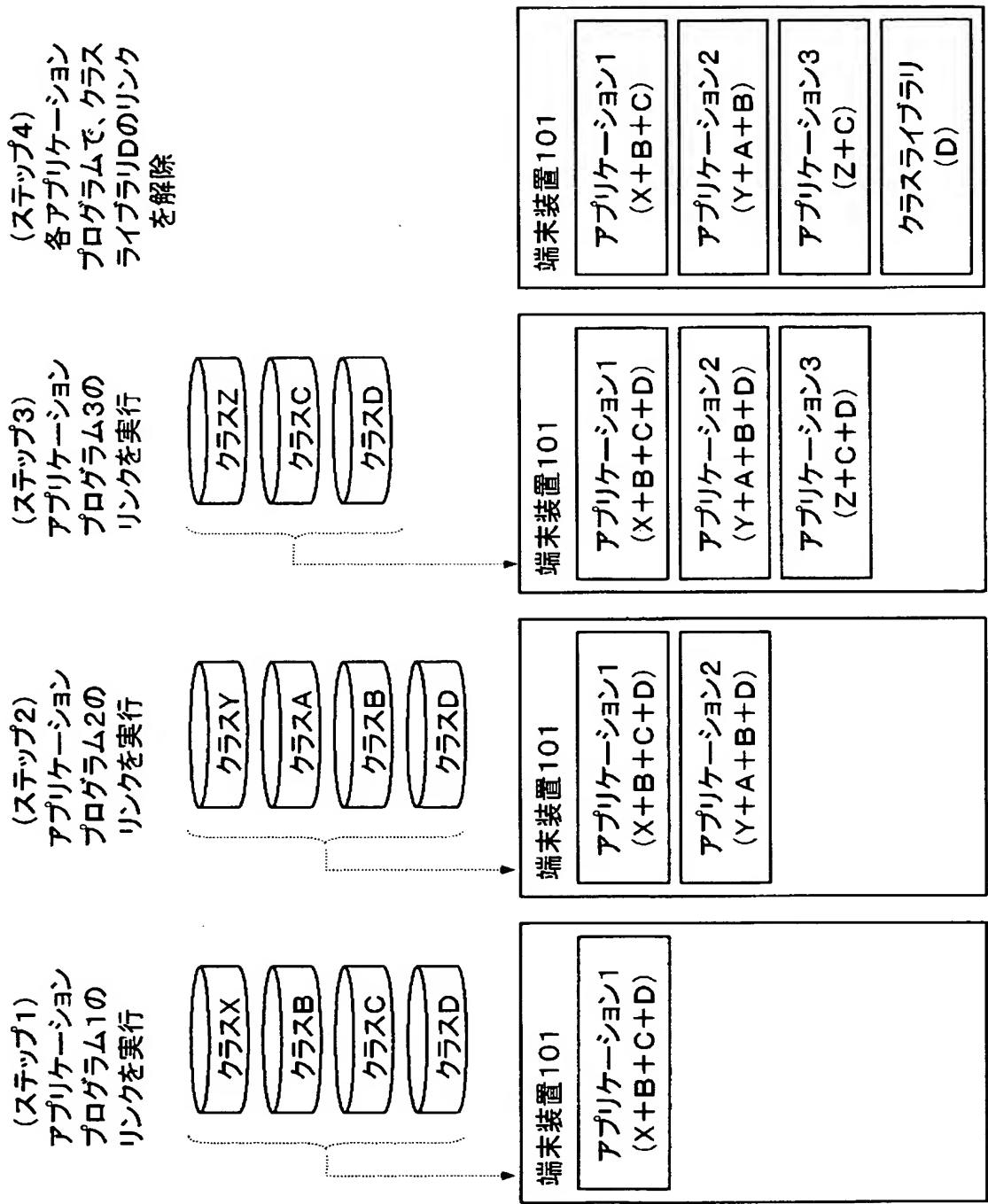
【図 2】



【図 3】



【図 4】



【図 5】

ステップ3

アプリケーション1 X+B+C+D	アプリケーション2 Y+A+B+D	アプリケーション3 Z+C+D	総クラスファイル数=11 実行時のリンク数=0
----------------------	----------------------	--------------------	----------------------------

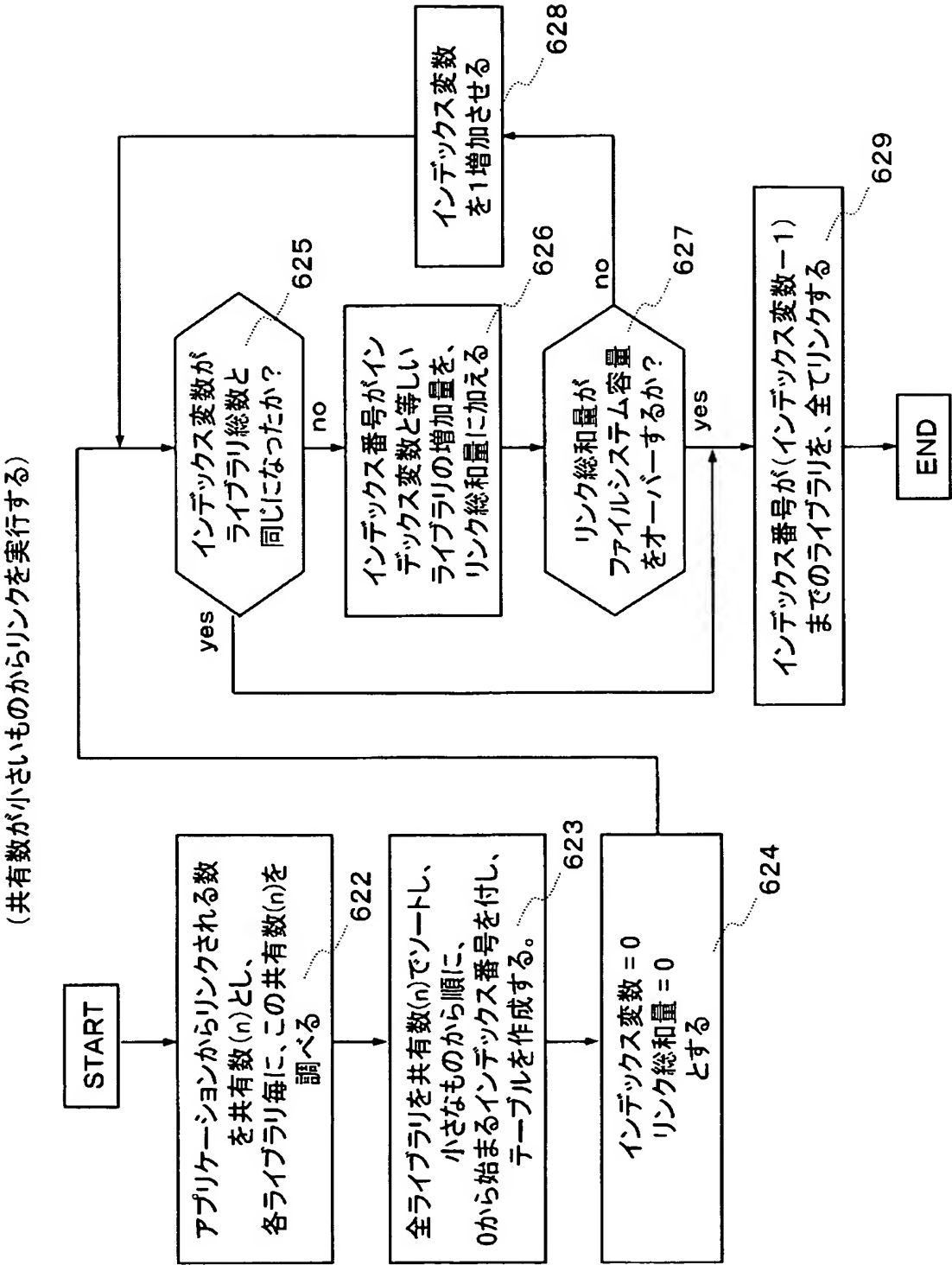
ステップ4

アプリケーション1 X+B+C	アプリケーション2 Y+A+B	アプリケーション3 Z+C	総クラスファイル数=9 実行時のリンク数=3
3個のアプリケーションで使用するクラスライブラリ D			

ステップ5

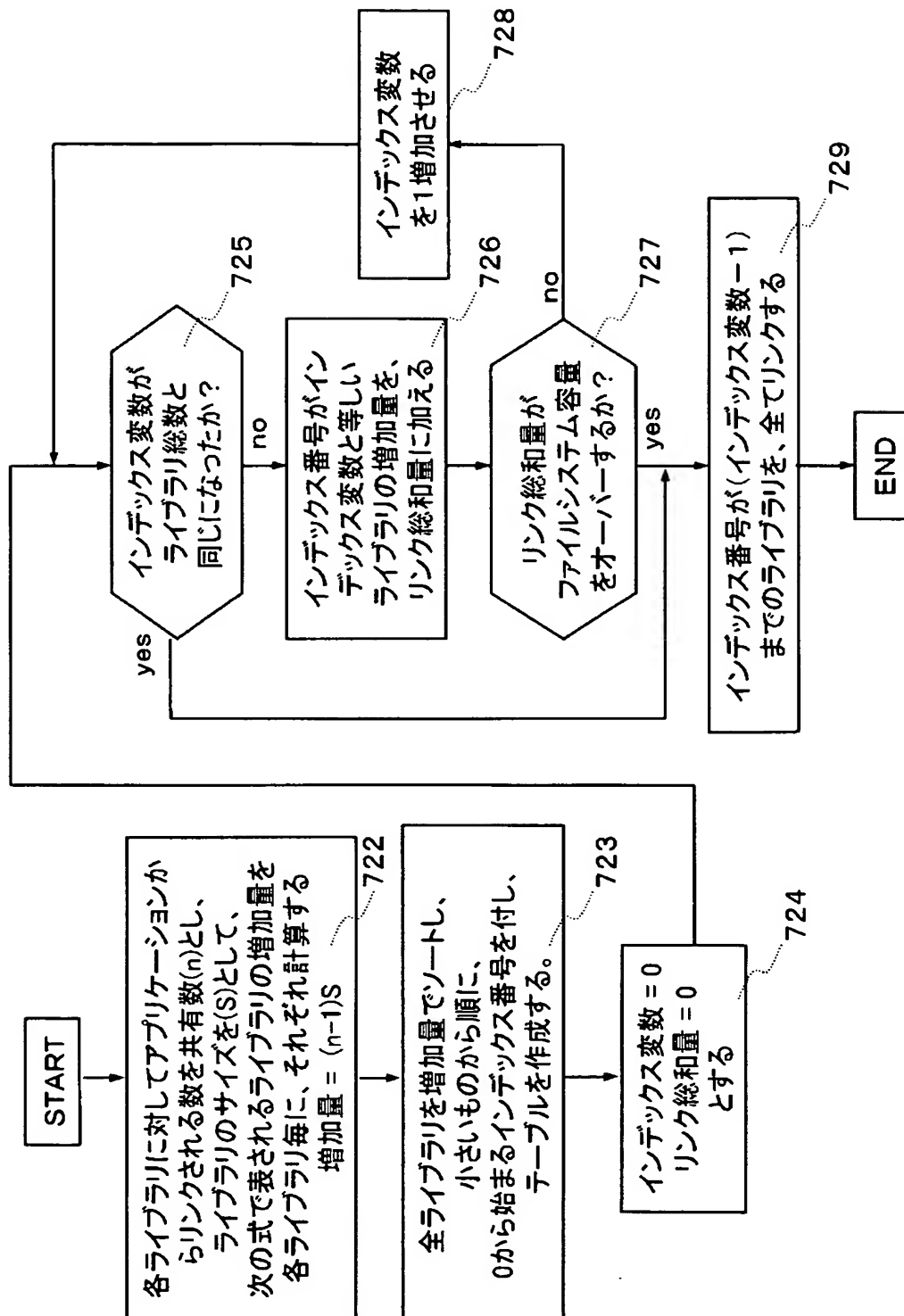
アプリケーション1 X	アプリケーション2 Y+A	アプリケーション3 Z	総クラスファイル数=7 実行時のリンク数=7
2個のアプリケーションで使用するクラスライブラリ B C			
3個のアプリケーションで使用するクラスライブラリ D			

【図 6】

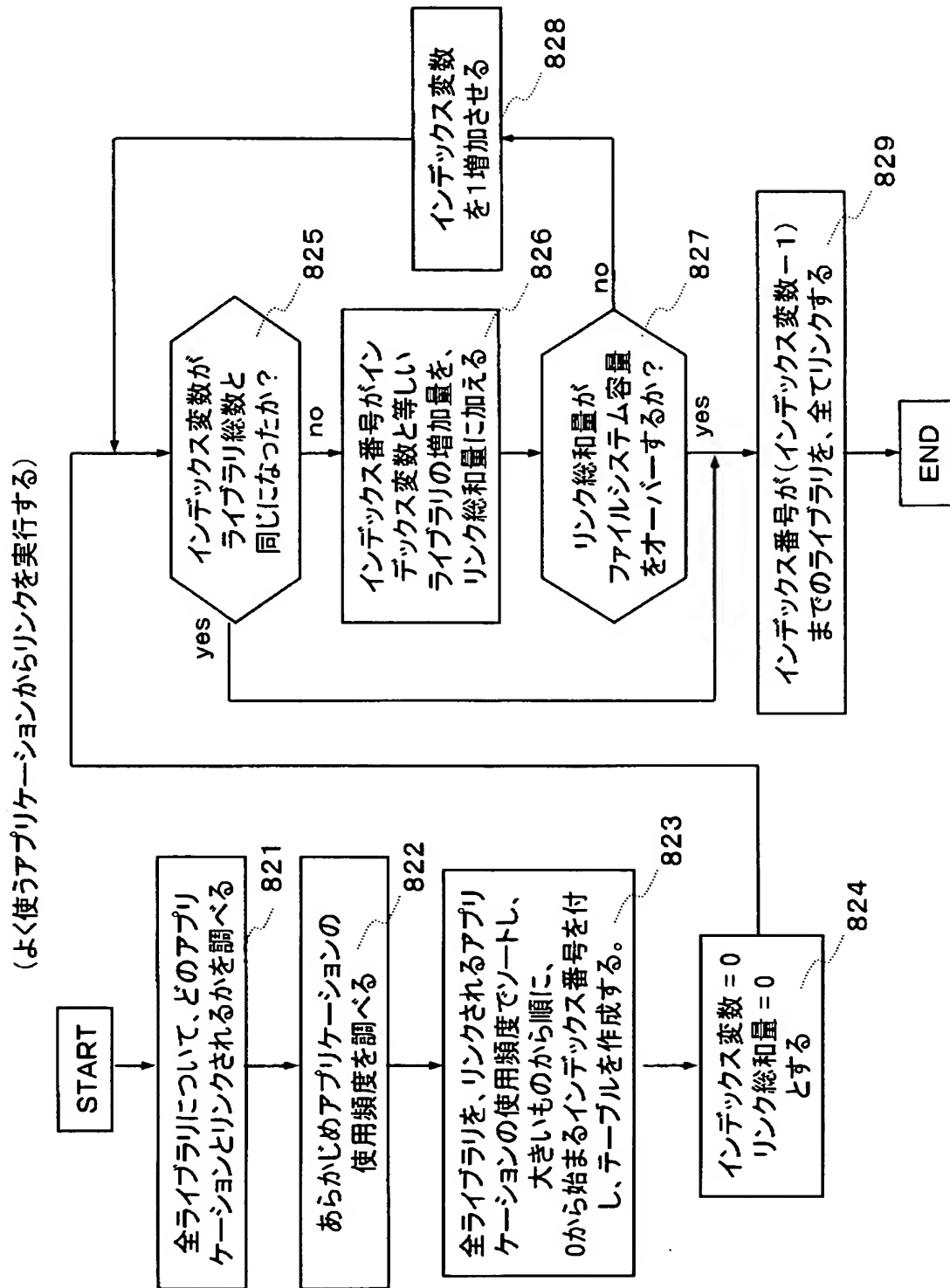


【図 7】

(メモリアイズの増加が小さいものからリンクを実行する)

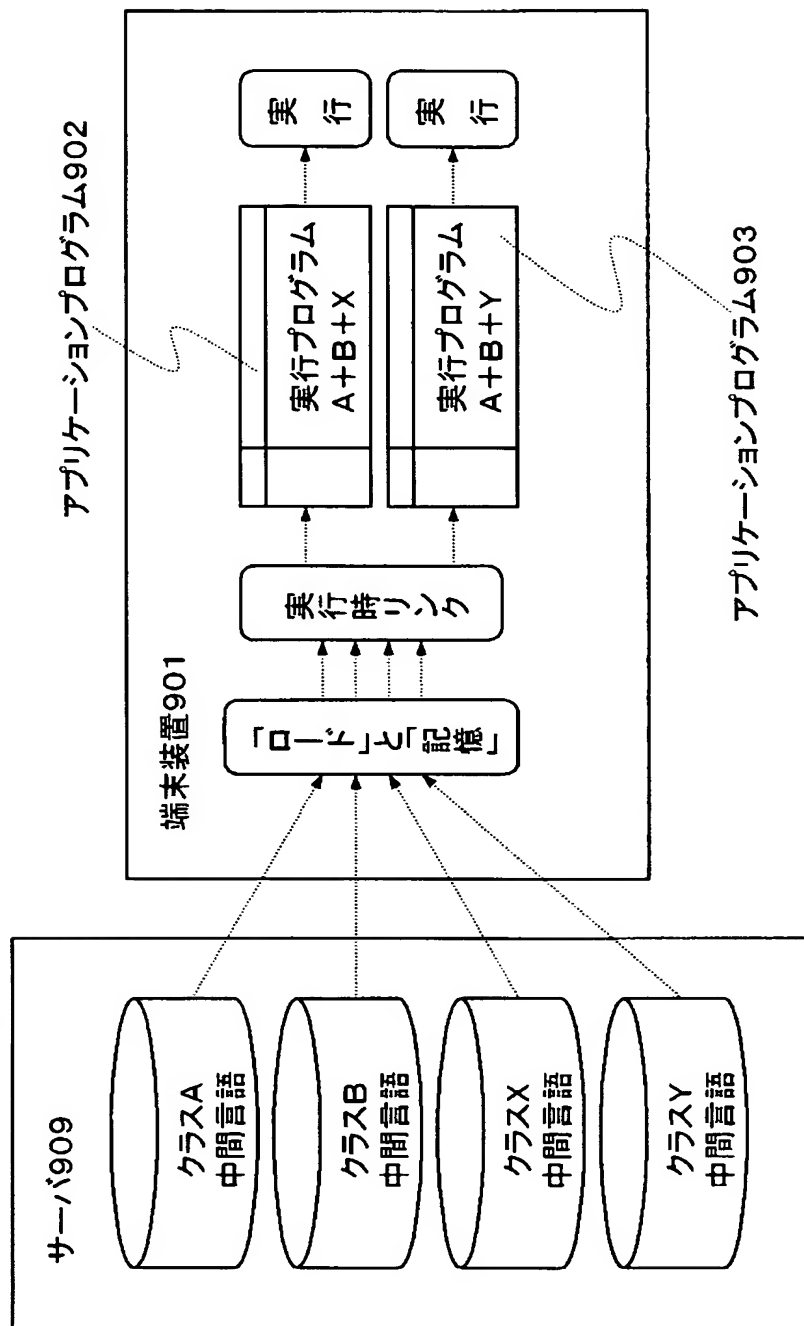


【図 8】



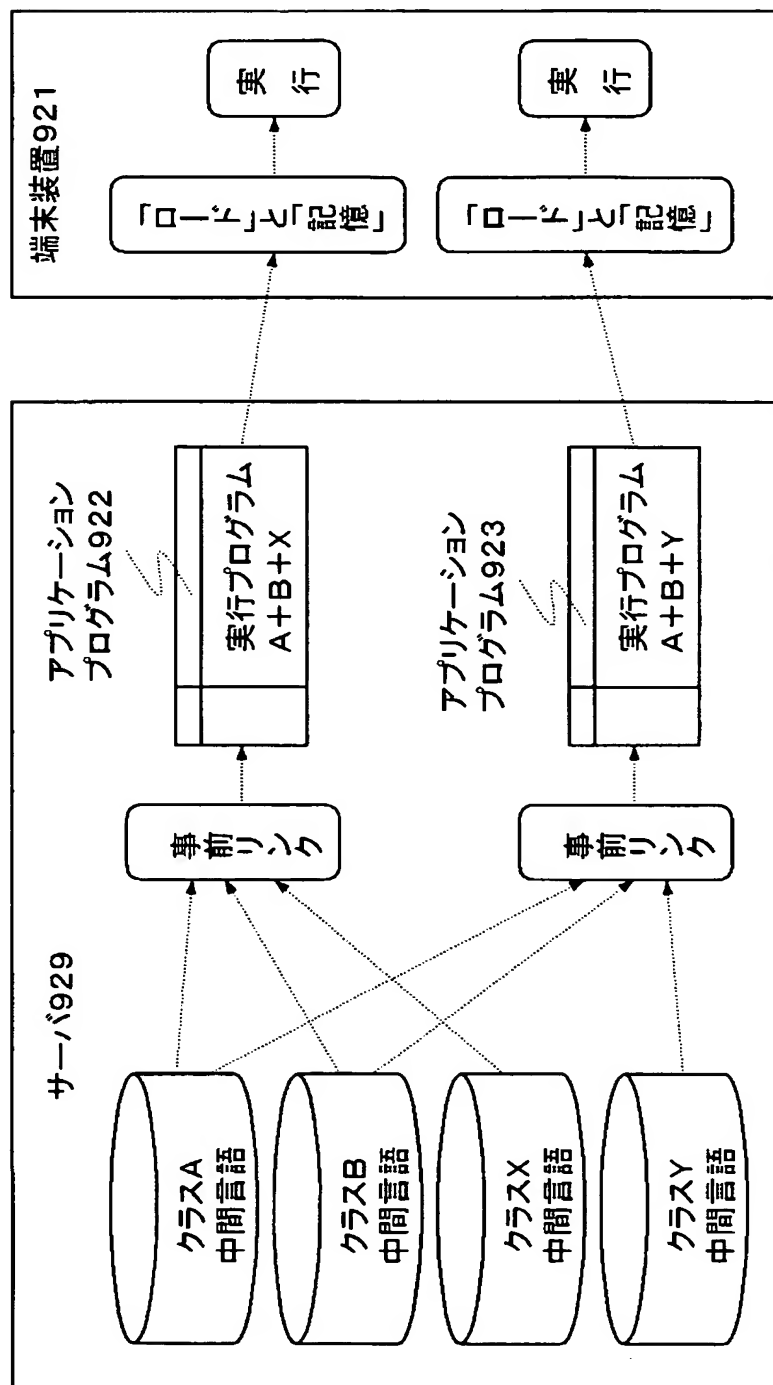
【図9】

従来のJava(R)リンカの実行例(実行時リンク方式)



【図10】

従来のCリンクの実行例(事前リンク方式)



【書類名】 要約書

【要約】

【課題】 限られたメモリ容量で、可能な限りライブラリを予めリンクしておくことにより、アプリケーション実行時の起動速度を改善する。

【解決手段】 複数のリンク前プログラムから 1 または複数個をリンクしてリンク後プログラムを作成し、前記リンク後プログラムを記憶手段に記憶し、前記記憶手段の使用状況を判断し、前記記憶手段の使用状況の判断に応じて前記リンク後プログラムを作成するために必要なリンクの一部だけを行うかあるいは全く行わないプログラムリンクプログラム。

【選択図】 図 1

特願 2 0 0 3 - 0 9 7 8 9 2

出 願 人 履 歴 情 報

識別番号

[0 0 0 0 0 5 8 2 1]

1. 変更年月日

1 9 9 0 年 8 月 2 8 日

[変更理由]

新規登録

住 所

大阪府門真市大字門真 1 0 0 6 番地

氏 名

松下電器産業株式会社